Declarative Integration of Interactive 3D Graphics into the World-Wide Web: Principles, Current Approaches, and Research Agenda

Jacek Jankowski*
Inria
Bordeaux
France

Sandy Ressler†
NIST
Gaithersburg MD
USA

Kristian Sons‡
DFKI
Saarbrücken
Germany

Yvonne Jung§
Fraunhofer IGD
Darmstadt
Germany

Johannes Behr¶
Fraunhofer IGD
Darmstadt
Germany

Philipp Slusallek‖
DFKI & Saarland University
Saarbrücken
Germany

Abstract

With the advent of WebGL, plugin-free hardware-accelerated interactive 3D graphics has finally arrived in all major Web browsers. WebGL is an imperative solution that is tied to the functionality of rasterization APIs. Consequently, its usage requires a deeper understanding of the rasterization pipeline. In contrast to this stands a declarative approach with an abstract description of the 3D scene. We strongly believe that such approach is more suitable for the integration of 3D into HTML5 and related Web technologies, as those concepts are well-known by millions of Web developers and therefore crucial for the fast adoption of 3D on the Web. Hence, in this paper we explore the options for new declarative ways of incorporating 3D graphics directly into HTML to enable its use on any Web page. We present declarative 3D principles that guide the work of the *Declarative 3D for the Web Architecture W3C Community Group* and describe the current state of the fundamentals to this initiative. Finally, we draw an agenda for the next development stages of Declarative 3D for the Web.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality I.3.6 [Methodology and Techniques]: Standards—Languages

Keywords: Declarative 3D, HTML5, DOM Integration, Polyfill

1 Introduction

The Web evolved from a text-based system to the current rich and interactive medium that supports images, 2D graphics, audio and video. These types of new media have made the Web experience richer, more attractive to users, etc, than ever before, and opened up possibilities for new types of applications and usage. The major media type that is still missing is 3D: synthetic, possibly photorealistic images in 3D with animation, as smoothly integrated in the everyday Web experience as images or video. Just as the appearance of images or video could open new application possibilities, access to the 3D on a Web site would make it possible to include realistic models of 3D objects – from models of buildings to representation of the human body or the sceneries for computer games. With WebGL [Marrin 2012], a JavaScript binding for OpenGL ES 2.0, this seems feasible; however, the goal would be to achieve the same smooth inclusion of 3D content in a Web page like we experience today with images or SVG-based 2D graphics.

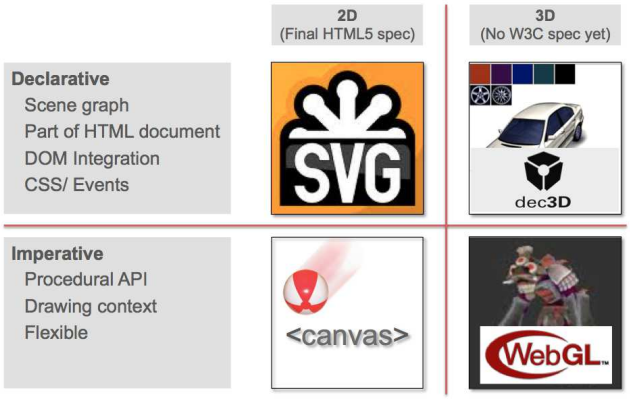

Figure 1: *The position of the Declarative 3D approach in the current Web graphics technology ecosystem.*

Although some of these goals could also be achieved by imperative means (e.g., through the usage of WebGL), developments of 3D models have a long tradition of using declarative approaches, which is also in line with some of the fundamental principles of Web development. It is therefore important to explore how the experiences accumulated in two different communities, namely the Web Development and Computer Graphics communities, can be capitalized upon to achieve the long term goal of using 3D on the Web the same way as we do with 2D graphics and video today. Moreover, while imperative graphics APIs are powerful and necessary, a *Declarative 3D* approach can provide web authors an easy way to add interactive high-level declarative 3D objects through the HTML Document Object Model (DOM) allowing them to easily create, modify, share, and experience interactive 3D graphics using HTML documents. Figure 1 depicts the position of the Declarative 3D approach in the current Web graphics technology ecosystem.

It is arguable that the emerging support for an imperative 3D API for the Web is useful but insufficient for broad acceptance and usage of 3D on the Web. A declarative approach that offers qualified concepts and that is tightly integrated with current web technologies, such as JSON or XML for scene construction, DOM for scene manipulation, and CSS for styling, is necessary to support a fast adoption and broad use of interactive 3D graphics by the millions of Web developers. The provided concepts must lift the hardware-oriented imperative application programming interfaces to an expressive and more easily usable level. Therefore, not the low-level

*e-mail:jacek.jankowski@inria.fr
†e-mail:sressler@acm.org
‡e-mail:kristian.sons@dfki.de
§e-mail:yvonne.jung@igd.fraunhofer.de
¶e-mail:johannes.behr@igd.fraunhofer.de
‖e-mail:slusallek@cs.uni-saarland.de

Final draft accepted to the 18th International Conference on 3D Web Technology (Web3D13), Jun 2013, San Sebastian, Spain

data structures of existing GPU layers must be in the center of the design but high-level elements and items like 3D objects, transformations, material descriptions, and lights. Instead of teaching Web developers 3D graphics APIs, the goal is to bring 3D graphics to the point where it is natural for Web developers to just make use of it. While this might not be feasible for every possible use of a low-level API, it still can cover the vast majority of use cases.

The *Declarative 3D for the Web Architecture W3C Community Group* (Dec3D) [W3C Community Group 2013] was thus founded to suggest and create methods to add high-level declarative 3D objects to the HTML DOM [W3C 2005], so users can share and experience interactive 3D contents. Moreover, this not only allows creating new content from existing content but also to index and search 3D content. The core mission is to determine the requirements, options, and use cases for the integration of interactive 3D graphics into the Web technology stack in a declarative way, which hopefully will provide a foundation for future standardization.

The group thus aims at presenting common use cases that define how 3D might intersect and interact with HTML5, DOM events, CSS, SVG, GeoLocation, Augmented Reality, Efficient XML Interchange (EXI) [W3C 2011] and other key working groups, whereas certain complex data types (e.g., transformation matrices) and computations are also of mutual interest. In this regard, this paper presents the current state and efforts concerning Dec3D.

2 Declarative 3D Principles

Here we describe declarative 3D principles, where the following goals should guide the development of DOM-based 3D graphics.

Following the Established Principles of the Web Declarative 3D is being developed to significantly lower the barrier for authoring 3D content for Web sites by duplicating the key features that enabled the growth of the early Web and its further success.

Separation of structure from content Underlying the Web from its earliest days was the separation of structure from content. The concept of a paragraph specified by the `<p>` tag was separate from its content. Declarative 3D is attempting to bring the same separation to 3D graphics inside of web pages. The concepts such as definitions of 3D objects, transformations, materials, etc. should be implemented in a declarative 3D description as an extension to HTML5 [Hickson 2012] using any existing or future extension mechanism.

Separation of content from style One of the principles of the current Web is also the separation of content and style, most notably through CSS. The successful integration of SVG [W3C 2012b] with HTML was made much easier due to the fact that SVG was already following this principle. The objective here is to extend the use of CSS for styling 3D graphics. One example can be the use of the latest CSS3 3D Transforms [W3C 2012a] to allow manipulating not only 2D but also 3D objects.

Use of the Document Object Model The DOM [W3C 2005] is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of Web documents. Declarative 3D should use the DOM API to examine and modify elements on the 3D scene and their attributes by simply reading and setting their properties. As the DOM provides access to user actions (e.g., pressing a key or clicking a mouse button), it should also be used as a main interface to interact with 3D contents.

Embedded 3D graphics should reuse existing W3C techniques, specifically from HTML5 and SVG, as far as possible and propose extensions only where specific features are necessary or provide significant benefits. Where new concepts are introduced their relation to and effects on existing Web standards should be analyzed, evaluated, and discussed with the respective W3C working groups.

3D Content Creation and Reuse While the creation of original 3D geometry and appearances still requires 3D specific know-how, the reuse, configuration, and manipulation of such content should be made similarly easy as for 2D Web content now. The solution should hide internal data structures and algorithms and provide users convenient ways to edit and manipulate such scenes. A key success factor for Declarative 3D on the Web will be the ability to generate new or reuse existing content. This requires that suitable exporters and converters can be built. However, as 3D on the Web is supposed mainly as a delivery mechanism, it is not necessary to include the ability to semantically represent all 3D features.

Platform Independence 3D content needs to be described in a way that does rely neither on a specific render API such as OpenGL or DirectX nor on a specific rendering technique such as rasterization or ray tracing only (cp. e.g. [Rubinstein et al. 2009; Schwenk et al. 2012]). This should allow for content to be portable across web browsers, rendering techniques, and hardware platforms, while taking advantage of available features wherever possible. The results of rendering content under such different environments should be highly predictable.

Efficiency and Scalability Interactive real-time 3D graphics enables new forms of interactivity on the Web but also adds significant new requirements on user agents. A key requirement for the selected technology therefore is the possibility to implement it efficiently (cf. e.g. [Trevett 2012]). Native implementations allow utilizing all (battery) resources more efficiently while leveraging heterogeneous hardware. Thereby, the CPU time can be used for the application instead for rendering, collision, scene-housekeeping, etc., which is esp. critical for mobile devices. Since 3D scenes can become rather large, any solution should target scalability in the sense that 3D content should run across different platforms (from mobile devices to high-end graphics hardware) with predictable performance. Mechanisms should be in place to handle cases where the performance provided by a user agent on some platform is not sufficient, e.g. by allowing for switching to different content (e.g. lower LOD) or provide alternate methods of delivering the content (e.g. server-based rendering delivered via streaming video).

Security and Digital Rights Management Secure delivery of Web content is a general problem and not specific to 3D data. However, the economic value of 3D data might make the problem more acute. Any proposed solution should therefore be based on a general approach to secure Web content. However, we first need to collect use cases, extract requirements and examine how far existing methods (e.g. [Koller and Levoy 2005]) and standards can be transferred to the proposed architecture. It is already demonstrated that the application of XML Encryption and Signature is needed for document fragments as well as full documents, since high-fidelity or sensitive portions of 3D models often need special protections.

Accessibility and Usability Accessibility improvements serve all users, not just people with disabilities. A problematic aspect of many 3D graphics approaches however is that user navigation and interaction is implemented inconsistently. Therefore, users familiar with one approach are impeded when navigating or interacting with other 3D scenes and models. Examination of relevant Web Accessibility Initiatives (WAI) principles might provide significant benefit.

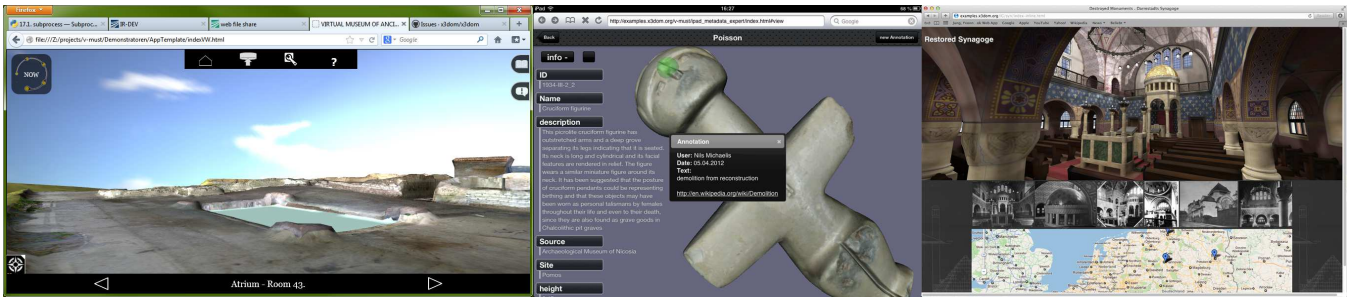

Figure 2: *From left to right: walktrough of Roman heritage site; cultural heritage object explorer with metadata; restored virtual synagoge.*

Conversely, use of declarative 3D graphics models might provide major benefits when describing the accessibility features and constraints of real-world objects and locations. Declarative 3D goals and potential solutions may achieve significant benefits if they are harmonizable with WAI imperatives.

Leveraging Web Development Infrastructure The Web has become the new application development platform for our time. Countless rich internet applications are being created at a dizzying pace. The "apps" world for mobile applications has also become a huge phenomenon. One important enabling technologies is JavaScript [Crockford 2008], which has evolved from a toy language into a robust and richly nuanced tool that is the basis for much of the recent explosion in Web applications. Declarative 3D, by living within the structure of Web apps is poised to leverage the huge collection of tools and infrastructure that already exists such as jQuery [Resig 2012] etc. The ability to take existing Web debuggers, editors, and viewers and use these tools to create, edit, and debug 3D content is a tremendous benefit and just begins to scratch the surface of development tools, created for Web application development, but which we can use for 3D content development.

3 Declarative 3D Frameworks

As already mentioned, the Declarative 3D Community Group [W3C Community Group 2013] has been formed to determine the requirements, options, and use cases for the declarative integration of interactive 3D graphics capabilities into the Web technology stack, which will provide a foundation for future standardization. While this standardization is our goal, we still need platforms allowing for the experimentation and evaluation of our design decisions (3.1). We also need to reach out to Web developers who could provide us with valuable feedback as early as possible. From the evaluations we extract the essential features (3.2) for a declarative approach. Finally, we check the level of integration we can achieve, emulating these features using existing technologies (3.3).

3.1 Evaluation platforms

With X3DOM [Behr et al. 2009] and XML3D [Sons et al. 2010], two evaluation platforms are available to support the ongoing discussion in the computer graphics and Web communities how an integration of HTML and declarative 3D content could look like.

Fraunhofer IGD's *X3DOM* [Behr et al. 2009; Behr et al. 2012b] is a JavaScript-based open-source framework for declarative 3D graphics in HTML5 that aims at extending the HTML DOM tree with declarative 3D objects while employing modern Web technologies like CSS3, Ajax, DOM scripting, as well as WebGL and – as fallback – Adobe's Flash 11 with Stage 3D [Adobe 2013] for GPU-accelerated rendering. The proposed 3D elements are mostly based upon the open ISO standard X3D [Web3D Consortium 2011], though X3DOM introduces a special HTML profile that basically extends the X3D Interchange profile. Additionally, instead of implementing the X3D pointing device sensor component, X3DOM simply uses, and appropriately extends, the HTML UI/Mouse events such that 3D pick events are likewise supported.

Furthermore, to overcome various problems that come along when embedding 3D mesh data, which typically consists of several megabytes of vertex attribute data, directly into the DOM tree, efficient 3D mesh encodings are being developed that allow separating the node structure from the raw vertex data [Behr et al. 2012b]. In this regard, Figures 2 and 7 show several X3DOM applications from the cultural heritage and engineering domain, where the ability to efficiently handle big 3D data sets is of high importance. Based on this work, 3D transmission formats are now also a major topic at the Khronos Group [Trevett 2012], where it is similarly argued that raw 3D data, just like image, video or audio data, needs to be externalized from the HTML document.

It is also worth mentioning that the Web applications shown in Figures 2 and 7 were not realized by graphics programmers, but mainly by Web developers using standard DOM scripting as well as JavaScript frameworks like jQuery [Resig 2012]. For example, to switch to a certain viewpoint when the user clicks onto the arrow button in the walkthrough application shown in Figure 2 (left), the application developer simply needs to update the 'bind' attribute of the "viewpoint" tag by calling its *setAttribute* method. Likewise, when selecting an element in the browser-based CAD and product structure viewer shown in the middle of Figure 7, the corresponding parts in the tree view are highlighted and vice versa.

XML3D [Sons et al. 2010] is a result of collaboration of DFKI and Intel VCI. Similarly to X3DOM, it employs CSS3, DOM scripting, and DOM events. In contrast to X3DOM, XML3D is designed as an extension to HTML5 from scratch. It does not define a scenegraph, but uses the DOM tree structure for parent child relations and second level references e.g. via CSS to (re-)use resources such as geometry data, material descriptions or animation data. These resources are not necessarily in the same document, but can also come from external documents in various formats, e.g. as JSON.

This generic and consistent handling of resources allows fine-granular composition of data sources. It can be combined with Xflow, a declarative data flow component [Klein et al. 2012] that allows for dynamic meshes, morphing, animation of shader parameters, image processing and Augmented Reality. The data flow computation can be mapped to hardware, e.g. by composition of WebGL shaders or by exploiting available APIs such as Web Workers [World Wide Web Consortium 2012] or River Trail [Herhut et al. 2012]. Figure 3 shows a XML3D scene with nine characters, skinned and animated with Xflow. There are two implementations of XML3D: the native implementation based on a modified

Figure 3: *A highly dynamic declarative 3D scene with nine animated and skinned characters. The calculation is either performed with classical sequential JavaScript or in parallel exploiting Intel's River Trail proposal for parallel data computations in JavaScript.*

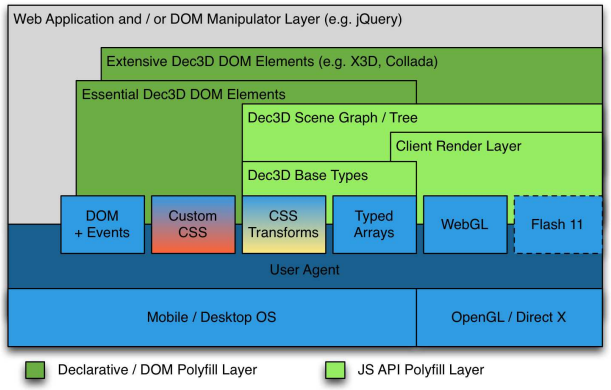

Figure 4: *The proposed Declarative 3D "Polyfill" Runtime Architecture.*

Chromium browser, and xml3d.js [Sons et al. 2013], a Polyfill implementation based on WebGL and JavaScript.

The X3DOM and XML3D platforms are available under an Open Source license and used in many national and international research and industrial projects. From the user feedback and use cases we have collected from both evaluation platforms, we were able to derive the essential features, that both approaches have in common and that we propose for a Dec3D standard.

3.2 Declarative 3D Essentials

The community group identified 15 key concepts of DOM-based Dec3D. In this section we briefly outline our proposed essential elements for an upcoming standard.

A *start element* (1) marks the transition point from HTML box layout to 3D transformations. This element itself is integrated in the flow layout, fully styleable with CSS and behaves similar to e.g. the `<canvas>` element. Within the 3D space we require a *hierarchy* (2) for 3D objects. HTML offers elements to structure documents hierarchically using elements such as `<div>`, `<span>`, `<section>`, etc. These elements could be reused. On the other hand, HTML5 recently extended the set of structural elements to allow a more granular definition of document semantics. It could be useful to do the same for 3D scenes. Still open is the question, if the reuse of sub-graphs should be supported. This is a common feature in X3D and can also be found in SVG. On the other hand, having multiple paths to one element introduces issues, e.g. with CSS inheritance.

CSS 3D Transforms [W3C 2012a] should be used to define *transformations* (3) on the hierarchical structure as well as on other 3D elements. This example illustrates the three following concepts:

```
<div>
  <dec3d style="border: 1px solid black;">
    <div style="transform: scale3d(2, 2, 2);">
      ...
    </div>
  </dec3d>
</div>
```

In general, a tight *CSS integration* (4) is a major concept: existing CSS properties should be used where applicable (e.g., transform, opacity, color, etc.). In addition, a set of 3D-specific CSS properties (e.g. for defining the appearance) needs to be defined. However, custom CSS properties are not yet supported by any Web browser, though finally this seems to change now [W3C 2013].

For real-time interactive 3D graphics systems an *event* (5) model is essential. The Dec3D standard should leverage the DOM Event model where applicable. Not only standard DOM UI Events such as 'mouseover' and 'click' (with appropriate extensions for 3D scenes as already discussed in [Behr et al. 2011]) shall be supported, but also new events need to be defined to provide 3D-specific context information. In this regard, the Browser API should expose convenience functions and special Dec3D *base types* (6) consisting of a lean set of complex data types for 2D/3D graphics. In general, we aim a common set of base types usable not only for Dec3D, but also for WebGL API, Audio API, SVG, 3D Transforms etc.

Similarly, *generic data containers* (7) should be based on TypedArrays [Khronos 2012] with appropriate interfaces for efficient modification of the data. These containers should be general enough to support meshes, animations, shaders, etc. This concept is very close to our efforts regarding *external data containers* (8) and their efficient transmission [Behr et al. 2012b; Trevett 2012]. Given that the generic data containers are close to vertex buffer objects in recent graphics APIs, we aim *3D geometry definitions* (9) close to OpenGL primitives.

Dec3D should support highly dynamic scenes. Using the DOM API to modify the scene data is not sufficient for many use cases, e.g. dynamic meshes, image processing, compression, etc. Thus, we need a concept to support efficient data processing that allows to map computations to available hardware. These should be expressed as *operators* (10) on the data containers above.

Besides this, we further distinguish between essential Dec3D DOM Elements, where the focus lies on a minimalistic element set, as well as more comprehensive Dec3D DOM Elements (see middle of Figure 4), which build on top of those with the focus on usability and which for convenience could build on existing scene-graph standards such as X3D [Web3D Consortium 2011] or COLLADA [Arnaud and Barnes 2006].

Essential DOM elements include a basic set of pre-defined *shaders* (11), which should be enough for typical use cases. Advantages are that they do not entail timing attack issues and that they can adapt to any target device. However, the design should not obstruct programmable shaders in the future. HTML elements such as `<img>`, `<video>`, and `<canvas>` can be used as *textures* (12), whereas elements such as `<svg>` and `<html>` could

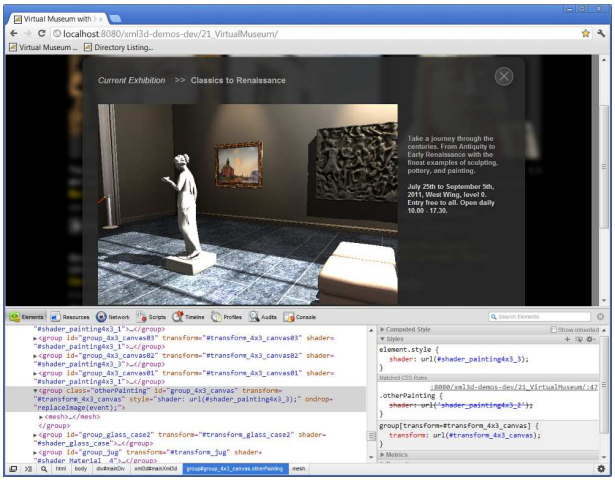

Figure 5: *LOI 3 in the native XML3D implementation: debugging of a 3D scene using the Chrome developer toolbar. Note also that materials are attached to meshes using the 'shader' CSS property.*

serve as interactive textures – once the corresponding security issues are resolved.

Lights (13) are shader parameters in OpenGL/ WebGL [Marrin 2012]. In contrast, for Dec3D we aim a representation to provide a intuitive way to define lights (and thereby also shadows) and to not obstruct more advanced future concepts. Likewise, *viewpoints* (14) can serve as links to certain points of interest within the document. However, it is still discussable if some pre-defined *camera navigation* (15) modes – to get directly started without having to modify the camera matrix with a special script – should be part of the essential profile.

3.3 Level of Integration and Polyfill Approach

We also propose a measure for the Level of Integration (LOI) for 3D graphics into the W3C technology stack (DOM, CSS, etc.) to explain existing and possible integration levels (Figure 6). Level 0 is the classical integration using plug-ins. Many X3D implementations as well as Adobe's Flash and Unity 3D use the plug-in model to integrate their rendering engine into the browser. LOI 1 provides a dedicated element in the DOM in combination with an API. Currently WebGL and the `<canvas>` provide this level of integration. The next level (LOI 2) has the scene description not in JavaScript but integrated in the DOM. Levels 3 and 4 provide an even better integration with CSS and easy debugging of 3D scenes. This requires having a set of 3D-specific CSS properties or even a full shader description via CSS as well as a deeper integration into debugging tools like Chrome Developer Tools or Firebug (for an example see Figure 5 and [Sons and Slusallek 2012]).

Using existing APIs in the browser, we can emulate native Dec3D functionality up to level 2, with some initial support of existing CSS properties. Integration levels 3 and 4 are currently not yet achievable by a JavaScript implementation, because the required APIs are missing: it is not possible to introduce new CSS properties or to extend developer tools by an API, a higher level of integration requires additional APIs in user agents. After discussions with browser vendors (namely Firefox and Chrome), who made it clear that integration of the desired extensions natively into their frameworks is not currently their priority, we decided to consider a so-called *Polyfill* approach for the further development activities. Nevertheless, communicating and requesting additional and missing features like custom CSS properties or CSS monitoring is necessary.

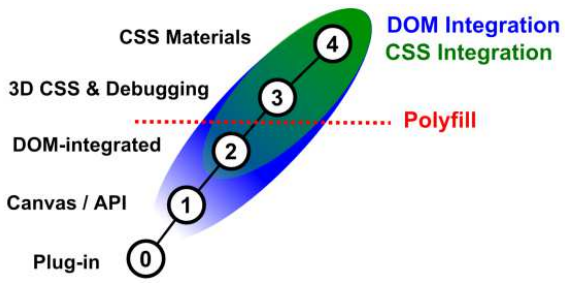

Figure 6: *Levels of integration: current browser APIs only allow integrating with DOM and DOM Events, but not yet fully with CSS.*

A *Polyfill* is a downloadable piece of code, which provides facilities that are not natively built-in to a Web browser [Sharp 2010]. For example, many features of HTML5 are not supported by versions of Internet Explorer older than version 8 or 9, but can be used by web pages if those pages install a so-called Polyfill. Polyfills can also be used to add entirely new functionality to browsers.

Polyfill-based approaches allow deriving hard requirements for related and utilized W3C standards and user agent (UA) APIs immediately. This leads to much more evolving concepts and solutions in contrast to an overall declarative 3D specification. In the following we hence list our most important UA requirements.

DOM The Polyfill Layer must be able to access and monitor changes in related DOM elements. This can be achieved using the now deprecated DOM Mutation Events as well as the new *MutationObserver*[1] objects.

Events The UA must support registration, firing, and extending UI events. This is mainly done through dynamic property changes in the JavaScript object, which currently represent the event for performance reasons.

Custom CSS properties The goal is to support scene management though custom CSS properties. Unfortunately, this was not possible in the last years, but may change with the new CSS Custom Properties specification [W3C 2013].

CSS 3D Transforms The CSS 3D Transforms module [W3C 2012a] is now supported in all major browsers, but the methods to monitor final matrix changes are limited right now and should be further extended for optimal performance.

TypedArrays TypedArrays [Khronos 2012] were first introduced with the WebGL specification but are now an established method to process large portions of typed data efficiently.

Generic GPU access For client-side rendering the Polyfill Layer has to access the GPU functionality almost directly. Here, WebGL is there a perfect candidate and Flash 11/Stage3D a second fallback option.

To sum up, the X3DOM and XML3D experimental declarative 3D Web publishing frameworks are designed to explore different options for adding 3D graphics to HTML. Here, Figure 4 depicts our proposed Declarative 3D Polyfill Runtime Architecture, though we would like to stress that the whole integration model is still evolving and open for discussion.

[1] `https://dvcs.w3.org/hg/domcore/raw-file/tip/Overview.html#mutation-observers`

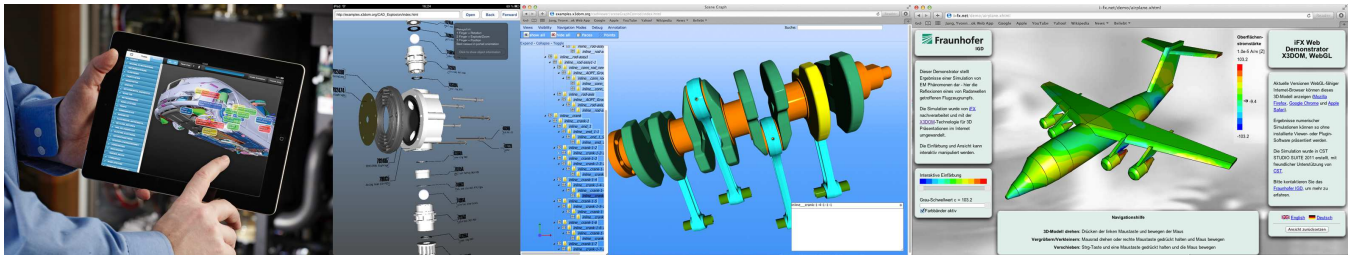

Figure 7: *From left to right: web-based design review application with annotation markers; interactive explosion of CAD model on iPad; lightweight web-based viewer for 3D CAD data; visualization and interactive exploration of simulation data by using extended mouse events.*

4 Declarative 3D Agenda

In this section we shape the agenda and identify upcoming research issues for the next development stage of Dec3D. During the 1st Intl. Workshop on Declarative 3D [Behr et al. 2012a] as well as during more informal meetings at Web3D and SIGGRAPH in 2011/2012, the members and supporters of the Declarative 3D W3C Community Group agreed upon the following topics to make the effort successful and for the W3C to adopt/develop a Dec3D standard.

Encourage Participation All relevant stakeholders, e.g. developers, designers, researchers, 3D artists, industry professionals, representatives of standards organizations, accessibility experts, and user-agent implementers, are encouraged to participate in this group. Participants should be willing to actively develop and donate materials towards the group's deliverables, as well as attend the group's teleconferences and face-to-face meetings.

Clear Definition of Use Cases and Requirements The group needs to agree on a collection of relevant use cases, where embedding 3D data in HTML using a declarative approach provides significant benefit. Here, declarative approaches esp. seem relevant for the industry, since they tend to think in formats (such as STL, STEP/IGES or CATIA) not in APIs. Each use case should explore how publishers and consumers benefit from Dec3D. From these use cases, the group needs to derive and prioritize the different required dimensions for the Dec3D technical specification [Jankowski 2012; Le Feuvre 2012].

Clear Technical Specification The next step is the creation of the clear, detailed and extensible technical specification of the implementation concepts and features necessary to cover a majority of useful requirements. Measurable properties need to be defined to quantitatively and/or qualitatively evaluate the achieved solution, document the pros and cons of each, and demonstrate that, based on the above analysis, there is a good chance of success in creating a W3C standard for Declarative 3D for the Web.

Outreach and Exemplar Applications The group needs to further continue its outreach activities through the high quality demonstrations of the Declarative 3D philosophy using the open-source frameworks X3DOM and XML3D. Therefore, several applications have to be identified, each requiring and demonstrating different capabilities of Dec3D. For example, one application could require huge 3D datasets, which are impractical for inclusion directly into the DOM; another could require real-time control of an external system; another could integrate with a complex data base; another an illustration of complex data (information visualization) without a physical analog, and yet another a more traditional scientific visualization. Key to selection of these applications is the use and demonstration of a more declarative 3D requirement.

W3C Working Group Proposal Finally, the Community Group should deliver reports documenting its progress, any conclusions it arrived at with respect to the standardization of *Declarative 3D for the Web* and, if reaching a positive conclusion, recommending a standardization approach as a basis for a future W3C working group on the same topic.

5 Conclusions

While WebGL, a 3D imperative graphics API in the Web context, is getting more and more traction, we are still missing an easy way to add interactive high-level declarative 3D objects to an HTML document to allow anyone to easily create, share, and experience interactive 3D graphics, with possibly wide ranging effects similar to those caused by the broad availability of video on the Web. The main motivation is thus to make it easy to add 3D graphics to Web pages by bringing 3D to the Web developers and not vice versa. This can be achieved by fully integrating 3D content into HTML5 documents, where interactive 3D graphics is a first class DOM objects. Moreover, by reusing existing Web technology wherever possible, no new concepts are added unless absolutely necessary.

Another objective of this position paper on Declarative 3D for the Web is to evaluate the options for a successful standardization of a declarative approach to interactive 3D graphics as part of HTML documents. The idea is to collect suitable use cases, derive requirements from them, and then find the essential set of features and concepts that enables broad uptake by authors and users of interactive 3D on the Web. We are absolutely aware that our goal is ambitious and it will take some time to implement these features. Therefore, we call for more participation from the Web3D and W3C communities that we believe is crucial to achieve our common and ultimate goal: 3D for everyone and everywhere.

Acknowledgements

The research presented in this paper has been supported by the EU projects V-Must (`http://v-must.net`), VERVE (`http://www.verveconsortium.eu`) and FI-CONTENT, the BMBF project EMERGENT, the Intel Visual Computing Institute, the SFI's Lion2 and EI's Copernicus (`http://copernicus.deri.ie`) projects, and the Villes Transparentes project.

References

ADOBE, 2013. Adobe flash player 11. http://www.adobe.com/products/flashplayer.html.

ARNAUD, R., AND BARNES, M. 2006. *Collada*. AK Peters.

BEHR, J., ESCHLER, P., JUNG, Y., AND ZÖLLNER, M. 2009. X3DOM – a DOM-based HTML5/ X3D integration model. In *Proc. Web3D 2009*, ACM Press, New York, USA, 127–135.

BEHR, J., JUNG, Y., DREVENSEK, T., AND ADERHOLD, A. 2011. Dynamic and interactive aspects of X3DOM. In *Proceedings Web3D 2011*, ACM Press, New York, USA, 81–87.

BEHR, J., BRUTZMAN, D., HERMAN, I., JANKOWSKI, J., AND SONS, K., Eds. 2012. Proceedings of the 1st Intl. Workshop on Declarative 3D for the Web Architecture (Dec3D2012 at WWW2012), vol. 869 of *CEUR Workshop Proceedings*.

BEHR, J., JUNG, Y., FRANKE, T., AND STURM, T. 2012. Using images and explicit binary container for efficient and incremental delivery of declarative 3d scenes on the web. In *Proc. Web3D*, ACM, New York, NY, USA, 17–25.

CROCKFORD, D. 2008. *JavaScript: The Good Parts*. O'Reilly.

HERHUT, S., HUDSON, R. L., SHPEISMAN, T., AND SREERAM, J. 2012. Parallel Programming for the Web. In *Proceedings of the 4th USENIX conference on Hot topics in parallelism*, USENIX Association, Berkeley, CA, USA, HotPar'12.

HICKSON, I., 2012. HTML5 (W3C Working Draft). http://www.w3.org/TR/2012/WD-html5-20120329/.

JANKOWSKI, J. 2012. Writing effective use cases for the declarative 3d for the web architecture. In *Dec3D'12*, CEUR-WS.

KHRONOS, 2012. Typed array spec. `http://www.khronos.org/registry/typedarray/specs/latest/`.

KLEIN, F., SONS, K., JOHN, S., RUBINSTEIN, D., SLUSALLEK, P., AND BYELOZYOROV, S. 2012. Xflow: declarative data processing for the web. In *Web3D*, ACM Press, 37–45.

KOLLER, D., AND LEVOY, M. 2005. Protecting 3d graphics content. *Commun. ACM 48*, 6, 74–80.

LE FEUVRE, J. 2012. Towards declarative 3d in web architecture. In *Dec3D'12*, CEUR-WS.

MARRIN, C., 2012. WebGL specification. `https://www.khronos.org/registry/webgl/specs/latest/`.

RESIG, J., 2012. jQuery. `http://jquery.com/`.

RUBINSTEIN, D., GEORGIEV, I., SCHUG, B., AND SLUSALLEK, P. 2009. RTSG: Ray Tracing for X3D via a Flexible Rendering Framework. In *Proceedings Web3D 2009*, ACM, New York, NY, USA, 43–50.

SCHWENK, K., JUNG, Y., VOSS, G., STURM, T., AND BEHR, J. 2012. CommonSurfaceShader revisited: Improvements and experiences. In *Proceedings Web3D 2012: 17th Intl. Conf. on 3D Web Technology*, ACM Press, New York, USA, 93–96.

SHARP, R., 2010. What is a polyfill? `http://remysharp.com/2010/10/08/what-is-a-polyfill/`.

SONS, K., AND SLUSALLEK, P. 2012. Demo: Xml3d interactive 3d graphics for the web. In *Dec3D'12*, CEUR-WS.

SONS, K., KLEIN, F., RUBINSTEIN, D., BYELOZYOROV, S., AND SLUSALLEK, P. 2010. Xml3d: interactive 3d graphics for the web. In *Web3D '10*, ACM.

SONS, K., SCHLINKMANN, C., KLEIN, F., RUBINSTEIN, D., AND SLUSALLEK, P. 2013. xml3d.js: Architecture of a Polyfill Implementation of XML3D. In *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2013 6th Workshop on*. to appear.

TREVETT, N. 2012. 3d transmission format. In *Seventh AR Standards Community Meeting Talks*.

W3C COMMUNITY GROUP, 2013. Declarative 3D for the Web Architecture. `http://www.w3.org/community/declarative3d/`.

W3C, 2005. Document Object Model (DOM). `http://www.w3.org/DOM/`.

W3C, 2011. Efficient xml interchange (exi) format. http://www.w3.org/TR/2011/REC-exi-20110310/.

W3C, 2012. Css 3d transforms. http://dev.w3.org/csswg/css3-3d-transforms/.

W3C, 2012. Svg. http://www.w3.org/Graphics/SVG/.

W3C, 2013. Css custom properties for cascading variables module level 1. http://dev.w3.org/csswg/css-variables/.

WEB3D CONSORTIUM, 2011. Extensible 3d (X3D). `http://www.web3d.org/x3d/specifications/`.

WORLD WIDE WEB CONSORTIUM, 2012. Web Workers – Editor's Draft. http://dev.w3.org/html5/workers/, March.